

## Case Study: Virtual Art Gallery

Team members: Elakkiya M, Machireddy Dhamini

## **Abstract**

This Virtual Art Gallery project is a console-based application developed using Python and MySQL to manage and display information about artworks, artists, users, and galleries. Instead of a graphical interface, the system uses simple text-based menus and commands to interact with users, making it easy to run on any computer without needing complex setups.

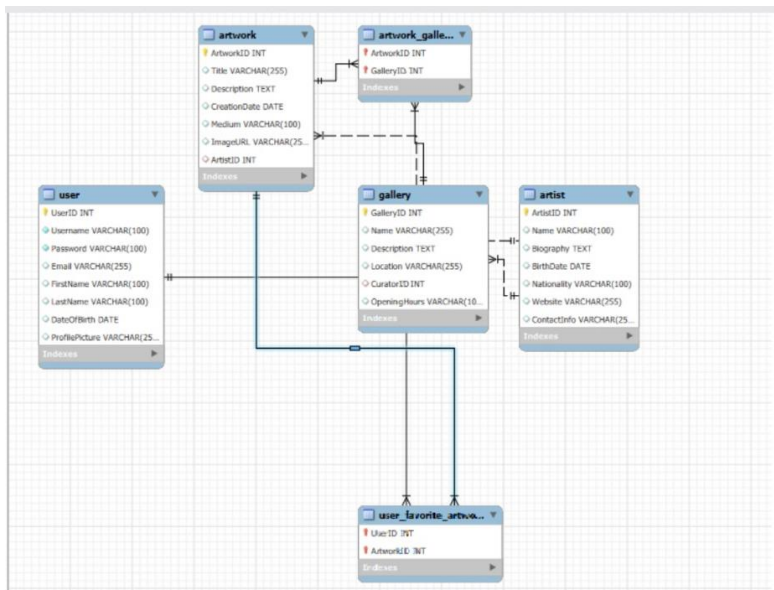
The project models key entities like Artwork, User, Artist, and Gallery using Python classes, which organize the data and actions related to each entity. The main functionalities include adding, viewing, and managing artwork details through console inputs, with all data stored securely in a MySQL database for easy retrieval and update.

By separating the program into different modules for database connection, business logic, and entity representation, the design remains clear and easy to maintain. Although the current version primarily focuses on artwork management, it lays a strong foundation to add more features for users, artists, and galleries in the future.

This console-based Virtual Art Gallery is a practical example of how programming and databases work together to build functional applications without a graphical interface. It allows users to manage and explore art collections through straightforward commands and outputs, providing a useful learning experience in database handling and object-oriented programming. Future improvements may include enhanced user interaction, data validation, and expanded features to better simulate a full art gallery system.

## Schema design:

- ER diagram:



- Created a dedicated database: VIRTUALARTGALLERY.

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| courier_management |
| elakkiya |
| hospital_management |
| information_schema |
| mysql |
| performance_schema |
| sakila |
| sys |
| virtualartgallery |
| world |
+-----+
10 rows in set (0.00 sec)
```

- All required tables were created with proper data types, keys, and relationships:

```
mysql> USE VIRTUALARTGALLERY;
Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_virtualartgallery |
+-----+
| artist                        |
| artwork                     |
| artwork_gallery              |
| gallery                     |
| user                        |
| user_favorite_artwork        |
+-----+
6 rows in set (0.00 sec)
```

Table Name	Description
ARTIST	Stores artist details
ARTWORK	Stores artwork information with reference to the artist
USER	Stores user profile details
GALLERY	Stores gallery data, each linked to a curator (artist)
USER_FAVORITE_ARTWORK	<b>Junction table</b> for many-to-many relationship between users and artworks
ARTWORK_GALLERY	<b>Junction table</b> for many-to-many relationship between artworks and galleries

All tables include primary keys, and necessary foreign key constraints

- Relationship Mapping (via Foreign Keys):

One-to-Many: ARTWORK → ARTIST, GALLERY → ARTIST

Many-to-Many: Handled via:

USER\_FAVORITE\_ARTWORK (USERID, ARTWORKID)

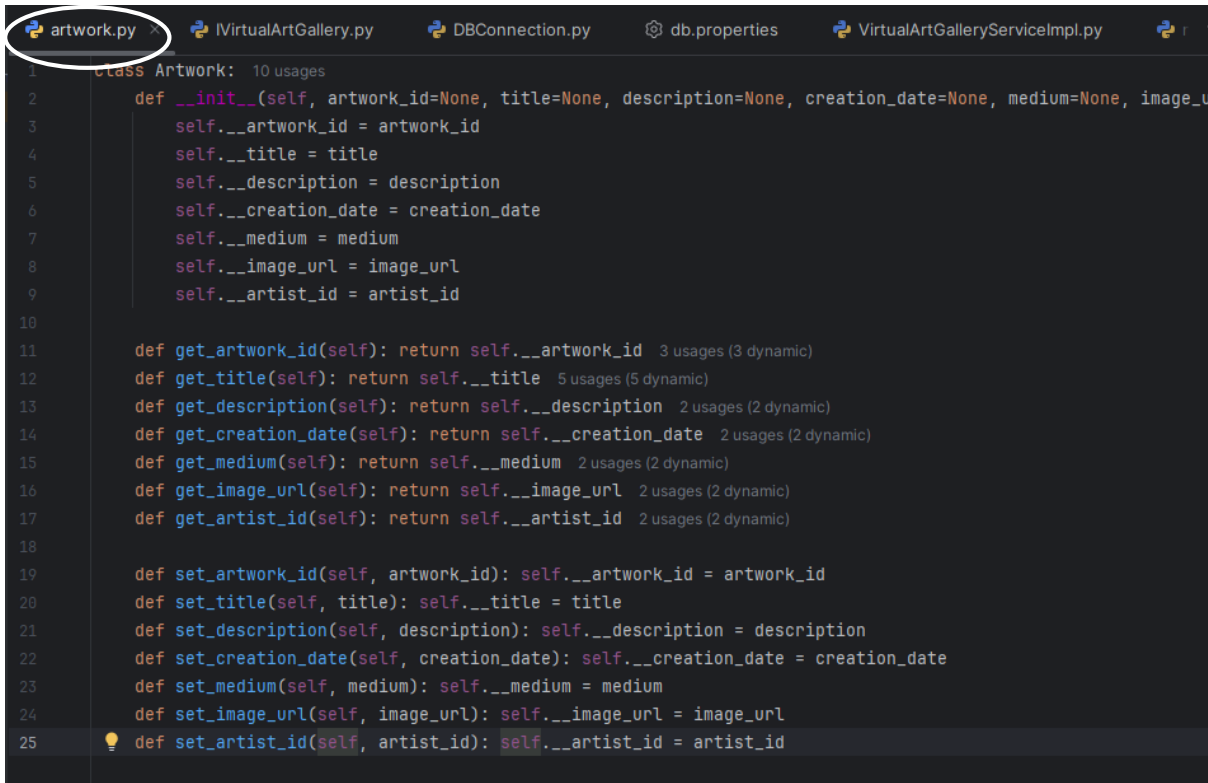
ARTWORK\_GALLERY (ARTWORKID, GALLERYID)

- Sample data inserted.

## Coding

### 1. Entity Classes

Created the model/entity classes corresponding to the schema within package entity with variables declared private, constructors(default and parametrized) and getters, setters )



```
1 class Artwork: 10 usages
2     def __init__(self, artwork_id=None, title=None, description=None, creation_date=None, medium=None, image_url=None, artist_id=None):
3         self.__artwork_id = artwork_id
4         self.__title = title
5         self.__description = description
6         self.__creation_date = creation_date
7         self.__medium = medium
8         self.__image_url = image_url
9         self.__artist_id = artist_id
10
11     def get_artwork_id(self): return self.__artwork_id 3 usages (3 dynamic)
12     def get_title(self): return self.__title 5 usages (5 dynamic)
13     def get_description(self): return self.__description 2 usages (2 dynamic)
14     def get_creation_date(self): return self.__creation_date 2 usages (2 dynamic)
15     def get_medium(self): return self.__medium 2 usages (2 dynamic)
16     def get_image_url(self): return self.__image_url 2 usages (2 dynamic)
17     def get_artist_id(self): return self.__artist_id 2 usages (2 dynamic)
18
19     def set_artwork_id(self, artwork_id): self.__artwork_id = artwork_id
20     def set_title(self, title): self.__title = title
21     def set_description(self, description): self.__description = description
22     def set_creation_date(self, creation_date): self.__creation_date = creation_date
23     def set_medium(self, medium): self.__medium = medium
24     def set_image_url(self, image_url): self.__image_url = image_url
25     def set_artist_id(self, artist_id): self.__artist_id = artist_id
```

Even though we use only artwork file for artwork management we also created other files in entity as per schema which can be used for future update

artwork.py User.py x Artist.py Gallery.py IVirtualArtGallery.py DBConnection.py db.prop

```

1 class User:
2     def __init__(self, user_id=None, username=None, password=None, email=None, first_name=None, last_name=None,
3         self.__user_id = user_id
4         self.__username = username
5         self.__password = password
6         self.__email = email
7         self.__first_name = first_name
8         self.__last_name = last_name
9         self.__dob = dob
10        self.__profile_picture = profile_picture
11
12    def get_user_id(self): return self.__user_id
13    def set_user_id(self, value): self.__user_id = value
14    def get_username(self): return self.__username
15    def set_username(self, value): self.__username = value
16    def get_password(self): return self.__password
17    def set_password(self, value): self.__password = value
18    def get_email(self): return self.__email
19    def set_email(self, value): self.__email = value
20    def get_first_name(self): return self.__first_name
21    def set_first_name(self, value): self.__first_name = value
22    def get_last_name(self): return self.__last_name
23    def set_last_name(self, value): self.__last_name = value
24    def get_dob(self): return self.__dob
25    def set_dob(self, value): self.__dob = value
26    def get_profile_picture(self): return self.__profile_picture
27    def set_profile_picture(self, value): self.__profile_picture = value

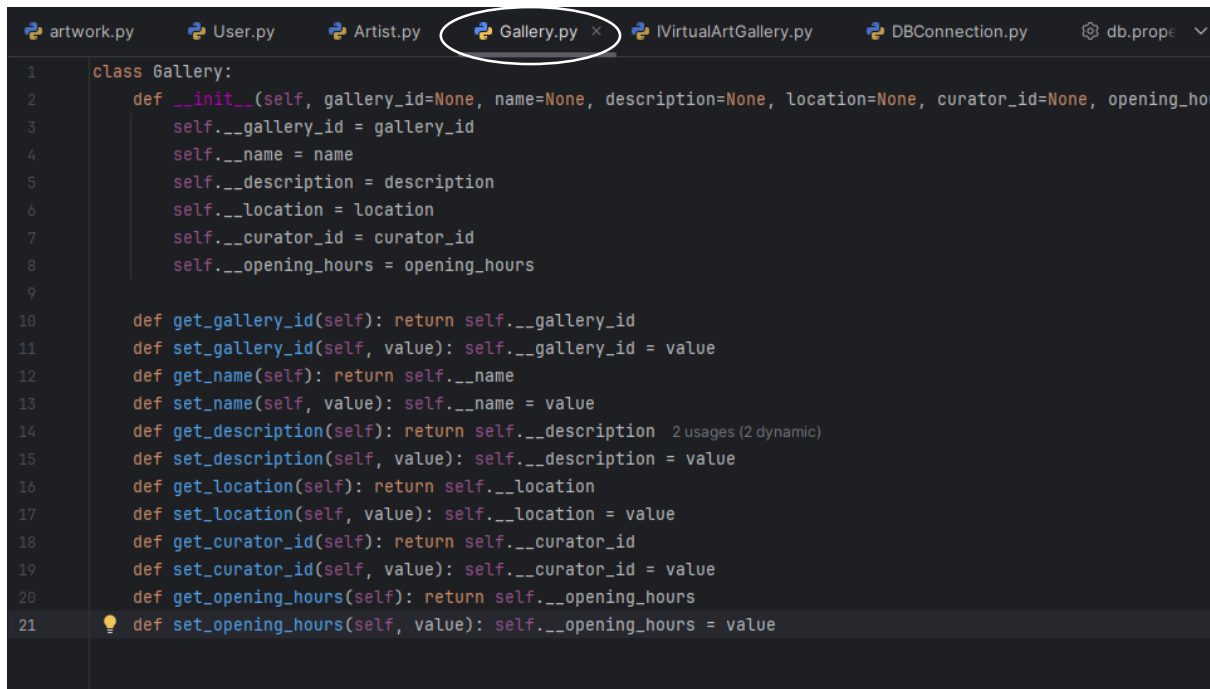
```

artwork.py User.py Artist.py x Gallery.py IVirtualArtGallery.py DBConnection.py db.prop

```

1 class Artist:
2     def __init__(self, artist_id=None, name=None, biography=None, birth_date=None, nationality=None, website=None,
3         self.__artist_id = artist_id
4         self.__name = name
5         self.__biography = biography
6         self.__birth_date = birth_date
7         self.__nationality = nationality
8         self.__website = website
9         self.__contact_info = contact_info
10
11    def get_artist_id(self): return self.__artist_id 2 usages (2 dynamic)
12    def set_artist_id(self, value): self.__artist_id = value
13    def get_name(self): return self.__name
14    def set_name(self, value): self.__name = value
15    def get_biography(self): return self.__biography
16    def set_biography(self, value): self.__biography = value
17    def get_birth_date(self): return self.__birth_date
18    def set_birth_date(self, value): self.__birth_date = value
19    def get_nationality(self): return self.__nationality
20    def set_nationality(self, value): self.__nationality = value
21    def get_website(self): return self.__website
22    def set_website(self, value): self.__website = value
23    def get_contact_info(self): return self.__contact_info
24    def set_contact_info(self, value): self.__contact_info = value
25

```

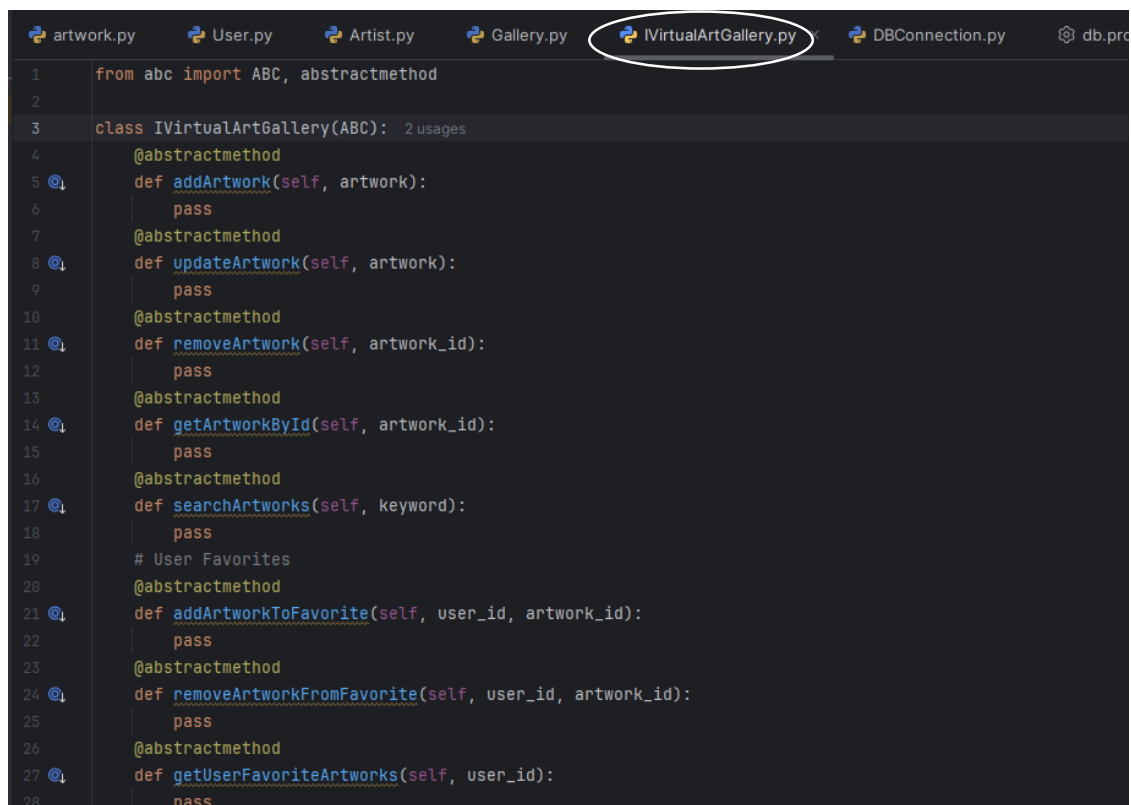


```
1 class Gallery:
2     def __init__(self, gallery_id=None, name=None, description=None, location=None, curator_id=None, opening_ho
3         self.__gallery_id = gallery_id
4         self.__name = name
5         self.__description = description
6         self.__location = location
7         self.__curator_id = curator_id
8         self.__opening_hours = opening_hours
9
10    def get_gallery_id(self): return self.__gallery_id
11    def set_gallery_id(self, value): self.__gallery_id = value
12    def get_name(self): return self.__name
13    def set_name(self, value): self.__name = value
14    def get_description(self): return self.__description 2 usages (2 dynamic)
15    def set_description(self, value): self.__description = value
16    def get_location(self): return self.__location
17    def set_location(self, value): self.__location = value
18    def get_curator_id(self): return self.__curator_id
19    def set_curator_id(self, value): self.__curator_id = value
20    def get_opening_hours(self): return self.__opening_hours
21    def set_opening_hours(self, value): self.__opening_hours = value
```

## 2. Interface Layer (dao/IVirtualArtGallery.py)

Defines abstract methods for:

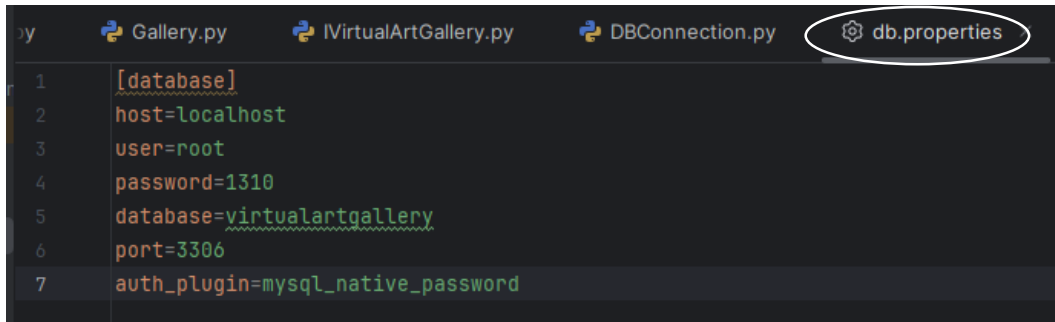
- Artwork management (add, update, delete, search)
- User favorite management (add/remove favorites, fetch list)



```
1 from abc import ABC, abstractmethod
2
3 class IVirtualArtGallery(ABC): 2 usages
4     @abstractmethod
5     def addArtwork(self, artwork):
6         pass
7     @abstractmethod
8     def updateArtwork(self, artwork):
9         pass
10    @abstractmethod
11    def removeArtwork(self, artwork_id):
12        pass
13    @abstractmethod
14    def getArtworkById(self, artwork_id):
15        pass
16    @abstractmethod
17    def searchArtworks(self, keyword):
18        pass
19    # User Favorites
20    @abstractmethod
21    def addArtworkToFavorite(self, user_id, artwork_id):
22        pass
23    @abstractmethod
24    def removeArtworkFromFavorite(self, user_id, artwork_id):
25        pass
26    @abstractmethod
27    def getUserFavoriteArtworks(self, user_id):
28        pass
```

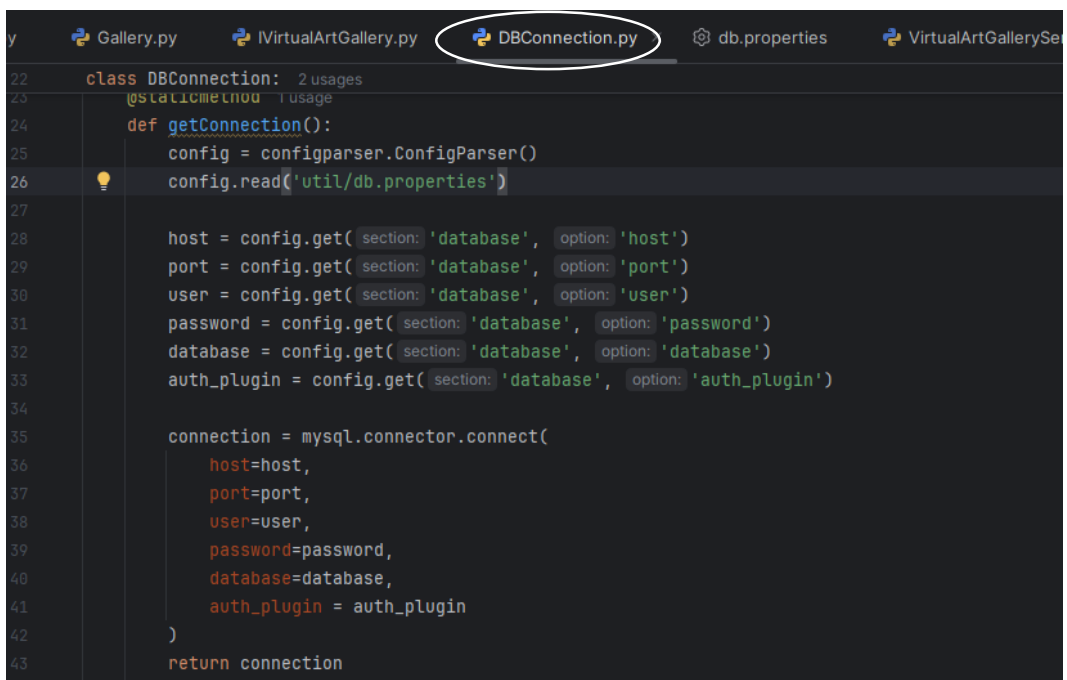
### 3. Database Utility (util/DBConnection.py)

- Reads MySQL credentials from db.properties
- Establishes and returns a MySQLConnection object



This screenshot shows the content of the `db.properties` file. The file is open in an IDE with tabs for `Gallery.py`, `IVirtualArtGallery.py`, `DBConnection.py`, and `db.properties` (the last one is circled). The code in `db.properties` is as follows:

```
1 [database]
2 host=localhost
3 user=root
4 password=1310
5 database=virtualartgallery
6 port=3306
7 auth_plugin=mysql_native_password
```



This screenshot shows the `DBConnection` class in `DBConnection.py`. The file is open in an IDE with tabs for `Gallery.py`, `IVirtualArtGallery.py`, `DBConnection.py` (circled), `db.properties`, and `VirtualArtGallerySe`. The code in `DBConnection.py` is as follows:

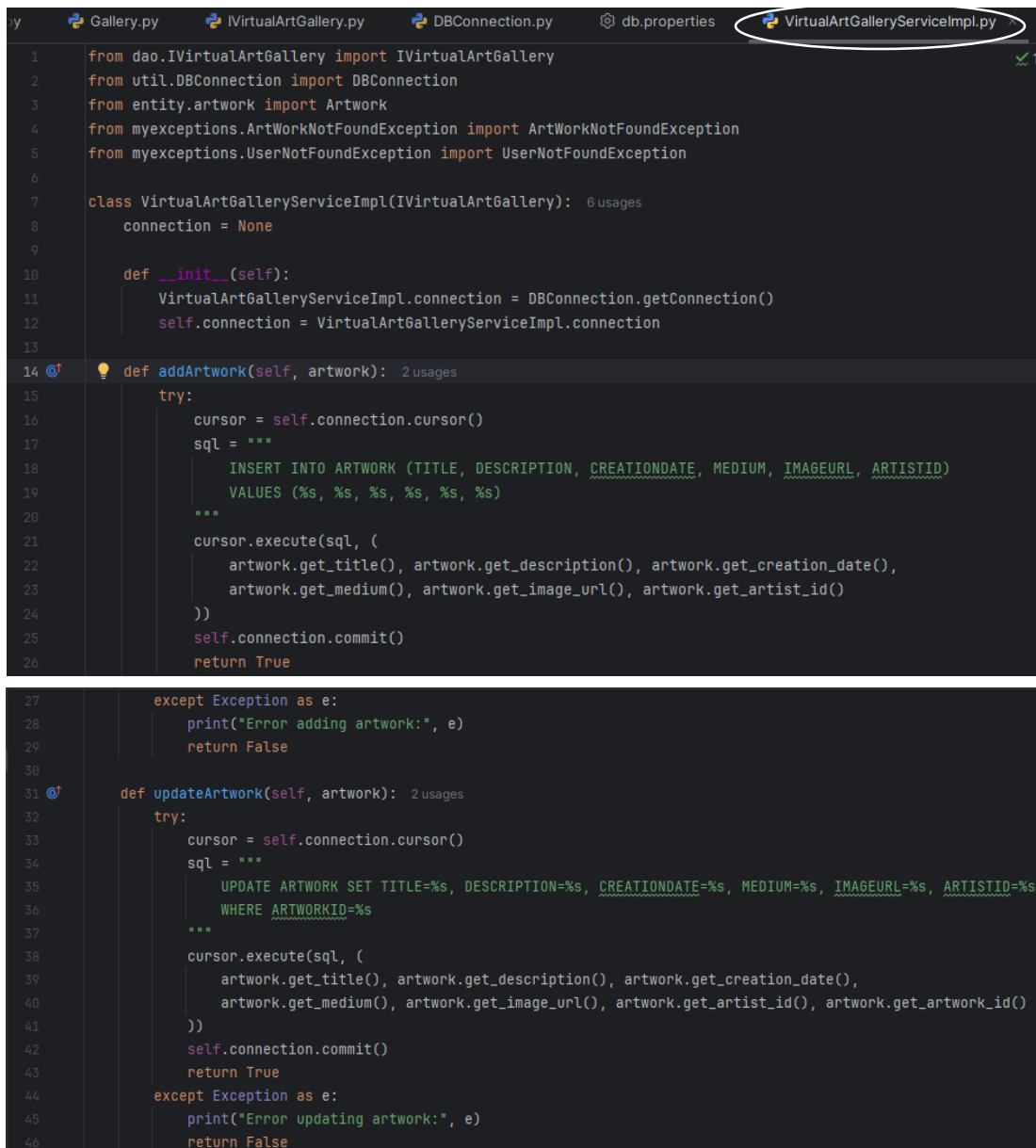
```
22 class DBConnection: 2 usages
23     @staticmethod 1 usage
24     def getConnection():
25         config = configparser.ConfigParser()
26         config.read('util/db.properties')
27
28         host = config.get(section='database', option='host')
29         port = config.get(section='database', option='port')
30         user = config.get(section='database', option='user')
31         password = config.get(section='database', option='password')
32         database = config.get(section='database', option='database')
33         auth_plugin = config.get(section='database', option='auth_plugin')
34
35         connection = mysql.connector.connect(
36             host=host,
37             port=port,
38             user=user,
39             password=password,
40             database=database,
41             auth_plugin = auth_plugin
42         )
43         return connection
```



## 4. Service Implementation (dao/VirtualArtGalleryServiceImpl.py)

Implements all interface methods with actual database operations:

- SQL queries (INSERT, SELECT, UPDATE, DELETE)
- Interacts with MySQL through a unified connection utility



```
1 from dao.IVirtualArtGallery import IVirtualArtGallery
2 from util.DBConnection import DBConnection
3 from entity.artwork import Artwork
4 from myexceptions.ArtWorkNotFoundException import ArtWorkNotFoundException
5 from myexceptions.UserNotFoundException import UserNotFoundException
6
7 class VirtualArtGalleryServiceImpl(IVirtualArtGallery): 6 usages
8     connection = None
9
10     def __init__(self):
11         VirtualArtGalleryServiceImpl.connection = DBConnection.getConnection()
12         self.connection = VirtualArtGalleryServiceImpl.connection
13
14     def addArtwork(self, artwork): 2 usages
15         try:
16             cursor = self.connection.cursor()
17             sql = """
18                 INSERT INTO ARTWORK (TITLE, DESCRIPTION, CREATIONDATE, MEDIUM, IMAGEURL, ARTISTID)
19                 VALUES (%s, %s, %s, %s, %s, %s)
20             """
21             cursor.execute(sql, (
22                 artwork.get_title(), artwork.get_description(), artwork.get_creation_date(),
23                 artwork.get_medium(), artwork.get_image_url(), artwork.get_artist_id()
24             ))
25             self.connection.commit()
26             return True
27
28         except Exception as e:
29             print("Error adding artwork:", e)
30             return False
31
32     def updateArtwork(self, artwork): 2 usages
33         try:
34             cursor = self.connection.cursor()
35             sql = """
36                 UPDATE ARTWORK SET TITLE=%s, DESCRIPTION=%s, CREATIONDATE=%s, MEDIUM=%s, IMAGEURL=%s, ARTISTID=%s
37                 WHERE ARTWORKID=%s
38             """
39             cursor.execute(sql, (
40                 artwork.get_title(), artwork.get_description(), artwork.get_creation_date(),
41                 artwork.get_medium(), artwork.get_image_url(), artwork.get_artist_id(), artwork.get_artwork_id()
42             ))
43             self.connection.commit()
44             return True
45         except Exception as e:
46             print("Error updating artwork:", e)
47             return False
```

```

47
48 def removeArtwork(self, artwork_id): 2 usages
49     try:
50         cursor = self.connection.cursor()
51         cursor.execute("DELETE FROM ARTWORK WHERE ARTWORKID = %s", (artwork_id,))
52         if cursor.rowcount == 0:
53             raise ArtWorkNotFoundException()
54         self.connection.commit()
55         return True
56     except Exception as e:
57         print("Error removing artwork:", e)
58         return False
59
60 def getArtworkById(self, artwork_id): 2 usages
61     try:
62         cursor = self.connection.cursor()
63         cursor.execute("SELECT * FROM ARTWORK WHERE ARTWORKID = %s", (artwork_id,))
64         row = cursor.fetchone()
65         if row:
66             return Artwork(*row)
67         else:
68             raise ArtWorkNotFoundException()
69     except Exception as e:
70         print("Error fetching artwork:", e)
71         raise
72
73 def searchArtworks(self, keyword): 2 usages
74     try:
75         cursor = self.connection.cursor()
76         cursor.execute("SELECT * FROM ARTWORK WHERE TITLE LIKE %s", (f"%{keyword}%",))
77         rows = cursor.fetchall()
78         return [Artwork(*row) for row in rows]
79     except Exception as e:
80         print("Error searching artworks:", e)
81         return []
82
83 def addArtworkToFavorite(self, user_id, artwork_id): 3 usages
84     try:
85         cursor = self.connection.cursor()
86         cursor.execute("INSERT INTO USER_FAVORITE_ARTWORK (USERID, ARTWORKID) VALUES (%s, %s)", (user_id, artwork_id,))
87         self.connection.commit()
88         return True
89     except Exception as e:
90         print("Error adding to favorites:", e)
91         return False
92
93 def removeArtworkFromFavorite(self, user_id, artwork_id): 2 usages
94     try:
95         cursor = self.connection.cursor()
96         cursor.execute("DELETE FROM USER_FAVORITE_ARTWORK WHERE USERID = %s AND ARTWORKID = %s", (user_id, artwork_id,))
97         if cursor.rowcount == 0:
98             raise ArtWorkNotFoundException("Artwork not found in user favorites")
99         self.connection.commit()
100         return True
101     except Exception as e:
102         print("Error removing from favorites:", e)
103         return False
104
105 def getUserFavoriteArtworks(self, user_id): 2 usages
106     try:
107         cursor = self.connection.cursor()
108         cursor.execute("SELECT * FROM USER_FAVORITE_ARTWORK WHERE USERID = %s", (user_id,))
109         if not cursor.fetchone():
110             raise UserNotFoundException()
111         cursor.execute("SELECT * FROM ARTWORK WHERE ARTWORKID IN (%s)", (','.join(str(row[1]) for row in cursor.fetchall()),))
112         rows = cursor.fetchall()
113         return [Artwork(*row) for row in rows]
114     except Exception as e:
115         print("Error fetching user favorite artworks:", e)
116         return []

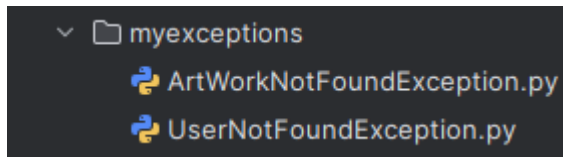
```

```

112         cursor.execute("""
113             SELECT A.* FROM ARTWORK A
114             JOIN USER_FAVORITE_ARTWORK UFA ON A.ARTWORKID = UFA.ARTWORKID
115             WHERE UFA.USERID = %s
116             """, (user_id,))
117         rows = cursor.fetchall()
118         return [Artwork(*row) for row in rows]
119     except Exception as e:
120         print("Error getting favorite artworks:", e)
121         return []
122

```

## 5. Exception Handling (myexceptions)



Custom exceptions:

- ArtWorkNotFoundException

The code editor shows the definition of the `ArtWorkNotFoundException` class. The file tab at the top is circled. The code is as follows:

```

1 class ArtWorkNotFoundException(Exception): 8 usages
2     def __init__(self, message="Artwork not found with the given ID"):
3         super().__init__(message)

```

- UserNotFoundException

The code editor shows the definition of the `UserNotFoundException` class. The file tab at the top is circled. The code is as follows:

```

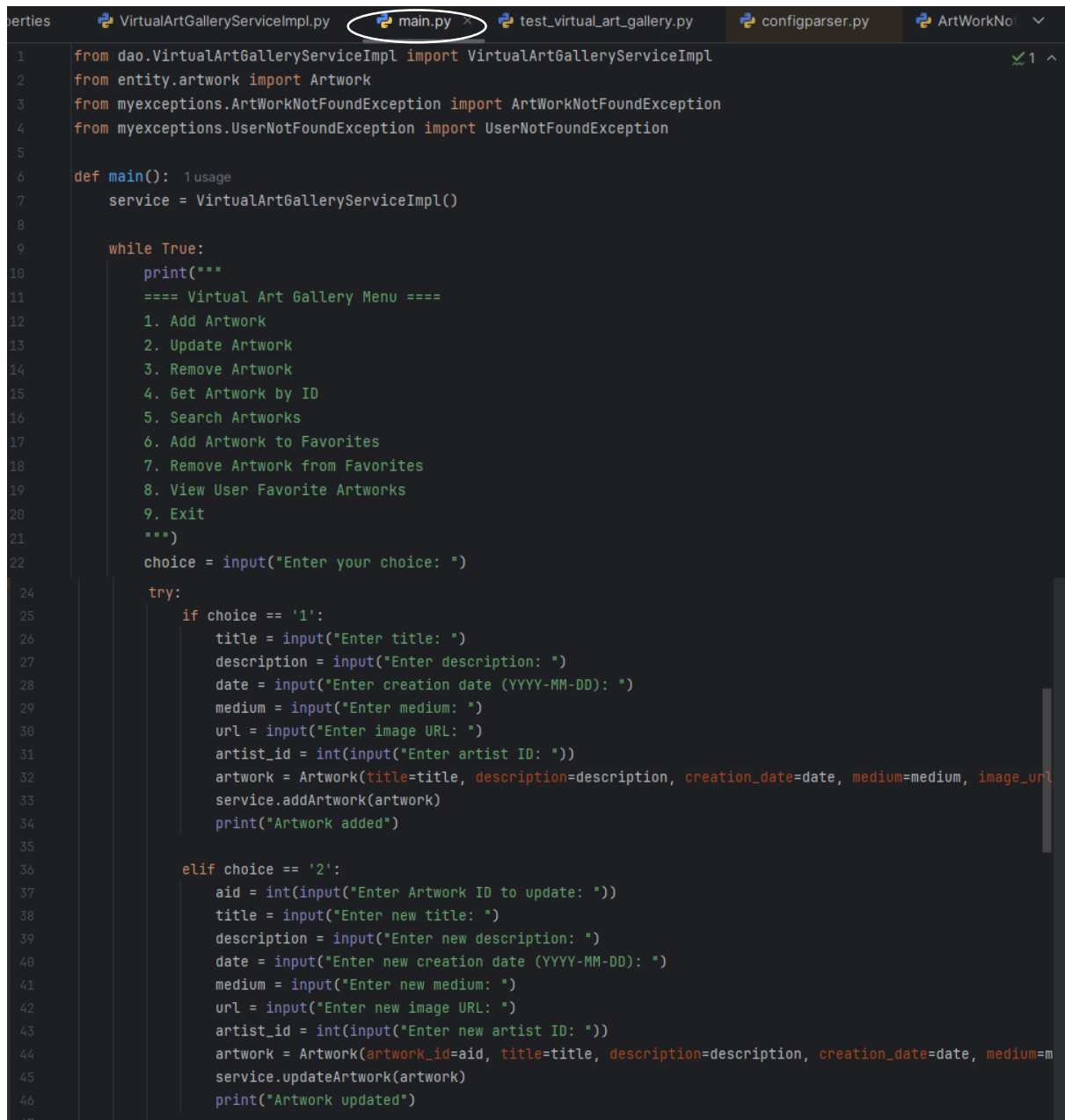
1 class UserNotFoundException(Exception): 6 usages
2     def __init__(self, message="User not found with the given ID"):
3         super().__init__(message)

```

## 6. Main Module (main/MainModule.py)

Menu-driven console program to:

- Trigger all services via user input
- Display results and error messages
- Loops back for continuous interaction



```
1 from dao.VirtualArtGalleryServiceImpl import VirtualArtGalleryServiceImpl
2 from entity.artwork import Artwork
3 from myexceptions.ArtWorkNotFoundException import ArtWorkNotFoundException
4 from myexceptions.UserNotFoundException import UserNotFoundException
5
6 def main(): 1 usage
7     service = VirtualArtGalleryServiceImpl()
8
9     while True:
10         print("""
11         ==== Virtual Art Gallery Menu ====
12         1. Add Artwork
13         2. Update Artwork
14         3. Remove Artwork
15         4. Get Artwork by ID
16         5. Search Artworks
17         6. Add Artwork to Favorites
18         7. Remove Artwork from Favorites
19         8. View User Favorite Artworks
20         9. Exit
21         """)
22         choice = input("Enter your choice: ")
23
24         try:
25             if choice == '1':
26                 title = input("Enter title: ")
27                 description = input("Enter description: ")
28                 date = input("Enter creation date (YYYY-MM-DD): ")
29                 medium = input("Enter medium: ")
30                 url = input("Enter image URL: ")
31                 artist_id = int(input("Enter artist ID: "))
32                 artwork = Artwork(title=title, description=description, creation_date=date, medium=medium, image_url=url, artist_id=artist_id)
33                 service.addArtwork(artwork)
34                 print("Artwork added")
35
36             elif choice == '2':
37                 aid = int(input("Enter Artwork ID to update: "))
38                 title = input("Enter new title: ")
39                 description = input("Enter new description: ")
40                 date = input("Enter new creation date (YYYY-MM-DD): ")
41                 medium = input("Enter new medium: ")
42                 url = input("Enter new image URL: ")
43                 artist_id = int(input("Enter new artist ID: "))
44                 artwork = Artwork(artwork_id=aid, title=title, description=description, creation_date=date, medium=medium, image_url=url, artist_id=artist_id)
45                 service.updateArtwork(artwork)
46                 print("Artwork updated")
47
```

```

48         elif choice == '3':
49             aid = int(input("Enter Artwork ID to remove: "))
50             service.removeArtwork(aid)
51             print("Artwork removed")
52
53         elif choice == '4':
54             aid = int(input("Enter Artwork ID: "))
55             try:
56                 art = service.getArtworkById(aid)
57                 print("Artwork Title:", art.get_title())
58             except ArtWorkNotFoundException as ae:
59                 print(ae)
60
61         elif choice == '5':
62             key = input("Enter keyword to search: ")
63             results = service.searchArtworks(key)
64             for art in results:
65                 print("ID:", art.get_artwork_id(), "Title:", art.get_title())
66
67         elif choice == '6':
68             uid = int(input("Enter User ID: "))
69             aid = int(input("Enter Artwork ID: "))
70             service.addArtworkToFavorite(uid, aid)
71             print("Favorite added")
72
73         elif choice == '7':
74             uid = int(input("Enter User ID: "))
75             aid = int(input("Enter Artwork ID: "))
76             service.removeArtworkFromFavorite(uid, aid)
77             print("Favorite removed")
78
79         elif choice == '8':
80             uid = int(input("Enter User ID: "))
81             try:
82                 favs = service.getUserFavoriteArtworks(uid)
83                 for art in favs:
84                     print("ID:", art.get_artwork_id(), "Title:", art.get_title())
85             except UserNotFoundException as ue:
86                 print(ue)
87
88         elif choice == '9':
89             print("Exiting...")
90             break
91         else:
92             print("Invalid choice")
93     except Exception as e:
94         print("Error:", e)
95 main()

```

Console output:

```

Run main
C:\Users\Admin\PycharmProjects\VIRTUAL_ART_GALLERY\.venv\Scripts\python.exe C:\Users\Admin\PycharmProjects\VIRTUAL_ART_GALLERY\main.py

==== Virtual Art Gallery Menu ====
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. View User Favorite Artworks
9. Exit

Enter your choice:

```

- Add artwork:

```

Enter your choice: 1
Enter title: kutrallam
Enter description: beauty of kutrallam falls
Enter creation date (YYYY-MM-DD): 2025-05-30
Enter medium: oil
Enter image URL: https://gallery.com/images/kutrallam.jpg
Enter artist ID: 4
Artwork added

```

```
mysql> SELECT * FROM ARTWORK;
```

ARTWORKID	TITLE	DESCRIPTION	CREATIONDATE	MEDIUM	IMAGEURL	ARTISTID
1	tanjai periya kovil	tanjaiyin azhagu	2024-12-18	oil	<a href="https://gallery.com/images/tanjai.jpg">https://gallery.com/images/tanjai.jpg</a>	2
2	Madurai Market	Flower market morning scene	2021-07-15	Watercolor	<a href="http://gallery.com/images/madurai.jpg">http://gallery.com/images/madurai.jpg</a>	1
3	Chennai Marina	Evening beach view	2023-03-22	Acrylic	<a href="http://gallery.com/images/marina.jpg">http://gallery.com/images/marina.jpg</a>	3
4	Coimbatore Hills	Western Ghats scenery	2020-08-05	Oil	<a href="http://gallery.com/images/coimbatore.jpg">http://gallery.com/images/coimbatore.jpg</a>	4
5	Kumbakonam Streets	Temple town street view	2023-02-18	Sketch	<a href="http://gallery.com/images/kumbakonam.jpg">http://gallery.com/images/kumbakonam.jpg</a>	5
6	Salem Fields	Sunset over rice fields	2022-09-09	Pastel	<a href="http://gallery.com/images/salem.jpg">http://gallery.com/images/salem.jpg</a>	6
7	Tirunelveli Traditions	Folk traditions of Tirunelveli	2021-11-11	Ink	<a href="http://gallery.com/images/tirunelveli.jpg">http://gallery.com/images/tirunelveli.jpg</a>	7
11	kutrallam	beauty of kutrallam falls	2025-05-30	oil	<a href="https://gallery.com/images/kutrallam.jpg">https://gallery.com/images/kutrallam.jpg</a>	4

8 rows in set (0.00 sec)

- Update artwork:

```

Enter your choice: 2
Enter Artwork ID to update: 11
Enter new title: kutrallam falls
Enter new description: beauty of kutrallam
Enter new creation date (YYYY-MM-DD): 2025-05-30
Enter new medium: oil
Enter new image URL: https://gallery.com/images/kutrallam.jpg
Enter new artist ID: 4
Artwork updated

```

The title and description is updated

```
mysql> SELECT * FROM ARTWORK;
```

ARTWORKID	TITLE	DESCRIPTION	CREATIONDATE	MEDIUM	IMAGEURL	ARTISTID
1	tanjai periya kovil	tanjaiyin azhagu	2024-12-18	oil	<a href="https://gallery.com/images/tanjai.jpg">https://gallery.com/images/tanjai.jpg</a>	2
2	Madurai Market	Flower market morning scene	2021-07-15	Watercolor	<a href="http://gallery.com/images/madurai.jpg">http://gallery.com/images/madurai.jpg</a>	1
3	Chennai Marina	Evening beach view	2023-03-22	Acrylic	<a href="http://gallery.com/images/marina.jpg">http://gallery.com/images/marina.jpg</a>	3
4	Coimbatore Hills	Western Ghats scenery	2020-08-05	Oil	<a href="http://gallery.com/images/coimbatore.jpg">http://gallery.com/images/coimbatore.jpg</a>	4
5	Kumbakonam Streets	Temple town street view	2023-02-18	Sketch	<a href="http://gallery.com/images/kumbakonam.jpg">http://gallery.com/images/kumbakonam.jpg</a>	5
6	Salem Fields	Sunset over rice fields	2022-09-09	Pastel	<a href="http://gallery.com/images/salem.jpg">http://gallery.com/images/salem.jpg</a>	6
7	Tirunelveli Traditions	Folk traditions of Tirunelveli	2021-11-11	Ink	<a href="http://gallery.com/images/tirunelveli.jpg">http://gallery.com/images/tirunelveli.jpg</a>	7
	<u>kutrallam falls</u>	<u>beauty of kutrallam</u>	2025-05-30	oil	<a href="https://gallery.com/images/kutrallam.jpg">https://gallery.com/images/kutrallam.jpg</a>	4

8 rows in set (0.00 sec)

- Remove artwork:

```
Enter your choice: 3
Enter Artwork ID to remove: 11
Artwork removed
```

```
mysql> SELECT * FROM ARTWORK;
```

ARTWORKID	TITLE	DESCRIPTION	CREATIONDATE	MEDIUM	IMAGEURL	ARTISTID
1	tanjai periya kovil	tanjaiyin azhagu	2024-12-18	oil	https://gallery.com/images/tanjai.jpg	2
2	Madurai Market	Flower market morning scene	2021-07-15	Watercolor	http://gallery.com/images/madurai.jpg	1
3	Chennai Marina	Evening beach view	2023-03-22	Acrylic	http://gallery.com/images/marina.jpg	3
4	Coimbatore Hills	Western Ghats scenery	2020-08-05	Oil	http://gallery.com/images/coimbatore.jpg	4
5	Kumbakonam Streets	Temple town street view	2023-02-18	Sketch	http://gallery.com/images/kumbakonam.jpg	5
6	Salem Fields	Sunset over rice fields	2022-09-09	Pastel	http://gallery.com/images/salem.jpg	6
7	Tirunelveli Traditions	Folk traditions of Tirunelveli	2021-11-11	Ink	http://gallery.com/images/tirunelveli.jpg	7

7 rows in set (0.00 sec)

The artwork ID 11 is removed

Exceptions:

```
Enter your choice: 3
Enter Artwork ID to remove: 8
Error removing artwork: Artwork not found with the given ID
```

- Get artwork by ID:

```
Enter your choice: 4
Enter Artwork ID: 2
Artwork Title: Madurai Market
```

- Search artwork:

```
Enter your choice: 5
Enter keyword to search: chennai
ID: 3 Title: Chennai Marina
```

- Add artwork to favourites:

```
Enter your choice: 6
Enter User ID: 1
Enter Artwork ID: 6
Favorite added
```

```
mysql> SELECT * FROM user_favorite_artwork;
```

USERID	ARTWORKID
2	1
1	2
3	2
1	3
3	3
4	4
2	5
5	5
1	6
6	6
7	7

11 rows in set (0.00 sec)

- Remove artwork from favourite:

```
Enter your choice: 7
Enter User ID: 1
Enter Artwork ID: 6

Favorite removed
```

```
mysql> SELECT * FROM user_favorite_artwork;
```

USERID	ARTWORKID
2	1
1	2
3	2
1	3
3	3
4	4
2	5
5	5
6	6
7	7

10 rows in set (0.00 sec)

The artwork is removed from favourite.

- View user favourite artworks:

```
Enter your choice: 8
Enter User ID: 1
ID: 2 Title: Madurai Market
ID: 3 Title: Chennai Marina
```



- Exiting:

```
==== Virtual Art Gallery Menu ====
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. View User Favorite Artworks
9. Exit

Enter your choice: 9
Exiting...

Process finished with exit code 0
```

## 7. Unit Testing (tests)

Tested key functionalities like:

- Adding/updating/removing artwork
  - Handling invalid IDs
  - Managing user favourites
- Uses Python's built-in unittest module.

```
Project
├── VIRTUAL_ART_GALLERY
│   ├── .venv
│   │   ├── library
│   │   └── root
│   ├── dao
│   ├── entity
│   ├── myexceptions
│   └── tests
│       └── test_virtual_art_gallery.py
│   ├── util
│   │   ├── db.properties
│   │   ├── DBConnection.py
│   │   └── main.py
│   └── External Libraries
│       └── Scratches and Consoles
└── DBConnection.py
    ├── db.properties
    ├── VirtualArtGalleryServiceImpl.py
    ├── main.py
    └── test_virtual_art_gallery.py

1 import unittest
2 from dao.VirtualArtGalleryServiceImpl import VirtualArtGalleryServiceImpl
3 from entity.artwork import Artwork
4 from myexceptions.ArtWorkNotFoundException import ArtWorkNotFoundException
5
6
7
8 class TestVirtualArtGallery(unittest.TestCase):
9
10     @classmethod
11     def setUpClass(cls):
12         cls.service = VirtualArtGalleryServiceImpl()
13
14     def test_add_artwork(self):
15         artwork = Artwork(title="Test Art", description="Test Desc", creation_date="2024-01-01", medium="Oil", 1)
16         result = self.service.addArtwork(artwork)
17         self.assertTrue(result)
18
19     def test_update_artwork(self):
20         artwork = Artwork(artwork_id=1, title="Updated Title", description="Updated Desc", creation_date="2024-01-01", medium="Oil", 1)
21         result = self.service.updateArtwork(artwork)
22         self.assertTrue(result)
23
```

Run Python tests for test\_virtual\_art\_gallery.TestVirtualArtG... x

Test Results 22 ms 11 tests passed 11 tests total, 22 ms

## **Conclusion:**

The Virtual Art Gallery project successfully demonstrates how Python can be used to build a console-based application for managing and exploring artworks, artists, users, and galleries. By applying object-oriented programming principles, the system models real-world entities in a structured and reusable way. Integration with a MySQL database allows for reliable data storage and retrieval.

Though the project does not include a graphical user interface, it effectively performs all core operations through a user-friendly command-line interface. Key functionalities such as adding, updating, removing, and searching artworks are implemented and tested. The system also supports features like managing user favourites, showcasing how basic user interaction can be handled using simple console outputs.

Unit testing ensures that the core services work correctly, while a clean separation of concerns in the code base (entities, services, exceptions, and utilities) makes the project maintainable and scalable. This project serves as a solid foundation for building more advanced applications in the future, such as a web-based or GUI-based virtual art gallery.

Overall, the project meets its primary goal of simulating a virtual art collection management system and provides valuable experience in Python programming, database connectivity and exception handling.